UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/647,106	08/25/2003	Andrew James Booker	550-462	9839

23117 7590 04/06/2007

NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
KHATRI, ANIL	
ART UNIT	**PAPER NUMBER**
2191	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/06/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/647,106	BOOKER ET AL.	
	Examiner	Art Unit	
	Anil Khatri	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 2/1/07.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-42 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-42 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-42 are rejected under 35 USC 101 because they disclose a claimed invention that is an abstract idea as defined in the case *In re Warmerdam*, 33, F 3d 1354, 31 USPQ 2d 1754 (Fed. Cir. 1994).

Examiner interprets that the claims 1-42 are non-statutory because claim is a computer program for processing set of instructions which capable of being executed by a computer implemented method, the computer program itself is not a process and without the computer-readable medium so its functionality can be realized. Applicant submit no substance that how this will be processed without incorporating a processor, memory and medium. Further, claims 1-42 are non-statutory because claim recites computer program product are program, per se i.e. the description or expressions of the program are not physical things nor are they statutory process as they do not act being performed. Computer programs do not define any structural and functional interrelationship between the computer program and other claimed aspect of the invention which permits the computer program's functionality could be realized. Therefore, computer program is merely a set of instructions capable of being executed by a computer, the computer program itself is not a process.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-42 are rejected under 35 U.S.C. 102(b) as being anticipated by *Cmelik et al* USPN 6,031,992.

Regarding claims 1, 14, 27 and 35

Cmelik et al teaches,

-generating, from a sequence of instructions, at least one of which includes a condition code, a corresponding sequence of generated instructions, for selected instructions having a condition code the corresponding generated instruction being a predetermined generated instruction having a corresponding condition code (column 4, lines 57-67," the host RISC processor and the hardware devices associated with it in a host RISC computer are usually quite different than are the devices associated with the processor for which the target application was designed; and the various instructions provided by the target application program are designed to cooperate with the device drivers of the target operating system in accessing the various portions of the target computer. Consequently, the emulation program, which changes the instructions of the target application program to primitive host instructions which the host operating system is capable of utilizing, must somehow link the operations designed to operate hardware devices in the target computer to operations which hardware devices of the host system are capable of implementing. Often this requires the emulator software to create virtual

devices which respond to the instructions of the target application to carry out operations which the host system is incapable of carrying out because the target devices are not those of the host computer. Sometimes the emulator is required to create links from these virtual devices through the host operating system to host hardware devices which are present but are addressed in a different manner by the host operating system);

executing, on a target processor, sequence of generated instructions and thereby producing software test information (column 5, lines 13-31, "target programs when executed in this manner run relatively slowly for a number of reasons. First, each target instruction from a target application program and from the target operating system must be changed by the emulator into the host primitive functions used by the host processor. If the target application is designed for a CISC machine such as an X86, the target instructions are of varying lengths and quite complicated so that changing them to host primitive instructions is quite involved. The original target instructions are first decoded, and the sequences of primitive host instructions which make up the target instructions are determined. Then the address (or addresses) of each sequence of primitive host instructions is determined, each sequence of the primitive host instructions is fetched, and these primitive host instructions are executed in or out of order. The large number of extra steps required by an emulator to change the target application and operating system instructions into host instructions understood by the host processor must be conducted each time an instruction is executed and slows the process of emulation) and

-when during step (b) predetermined generated instruction is encountered, determining operation of target processor whether the condition code of predetermined generated instruction is satisfied and, if so, replacing predetermined generated instruction with

corresponding instruction from sequence of instructions so as to cause corresponding instruction to be executed (column 11, lines 54-67, "the code morphing software combined with the enhanced morph host translates target instructions into instructions for the morph host on the fly and caches those host instructions in a memory data structure (referred to in this specification as a "translation buffer"). The use of a translation buffer to hold translated instructions allows instructions to be recalled without rerunning the lengthy process of determining which primitive instructions are required to implement each target instruction, addressing each primitive instruction, fetching each primitive instruction, optimizing the sequence of primitive instructions, allocating assets to each primitive instruction, reordering the primitive instructions, and executing each step of each sequence of primitive instructions involved each time each target instruction is executed. Once a target instruction has been translated, it may be recalled from the translation buffer and executed without the need for any of these myriad of steps).

Regarding claims 2, 15 and 28

Cmelik et al teaches,

each instruction of said sequence of instructions includes a condition code (column 19, lines 42-51, "he manner in which the primitive instruction which precedes the branch instruction may update the value of the instruction pointer for the target processor is to test the condition code for the branch in the condition code registers and then determine whether one of the two branch addresses indicated by the condition controlling the branch is stored in the translation buffer 14. The first time the sequence of target instructions is translated, the two branch targets of the host instruction both hold the same host processor address for the main loop of the translator

software. When the host translation is completed, stored in the translation buffer 14, and executed for the first time, the instruction pointer is updated in the target instruction pointer register (as are the rest of the target registers); and the operation branches back to the main loop. At the main loop, the translator software looks up the instruction pointer to the next target instruction in the target instruction pointer register).

Regarding claims 3, 16, 29 and 36

Cmelik et al teaches,

condition code is an instruction qualifier, which prevents the instruction from being executed by target processor unless status information satisfies condition code (column 20, When this condition is reached, the time required to fetch target instructions, decode target instructions, fetch the primitive instructions which make up the target instructions, optimize those primitive operations, reorder the primitive operations, and reschedule those primitive operations before running any host instruction is eliminated. Thus, in contrast to all prior art microprocessors which must take each of these steps each time any application instruction sequence is run, the work required to run any set of target instructions using the present invention after the first translation has taken place is drastically reduced. This work is further reduced as each set of translated host instructions is linked to the other sets of translated host instructions. In fact, it is estimated that translation will be needed in less than one translation execution out of one million during the running of an application).

Regarding claims 4, 17 and 37

Cmelik et al teaches,

status information is predetermined architectural state associated with target processor and condition code specifies a status of predetermined architectural state that must be met in order for the instruction to be executed (column 20, lines 5-15, "second translated... buffer).

Regarding claims 5, 13, 18, 26, 30, 38 and 42

Cmelick et al teaches,

predetermined generated instruction is an instruction, which is not recognized by target processor (column 7, lines 22-45, The target software run in this manner runs relatively slowly for the same reasons that the emulation of FIG. 1(b) runs slowly. First, each target instruction from the target application and from the target operating system must be changed by fetching the instruction; and all of the host primitive functions derived from that instruction must be run in sequence each time the instruction is executed. Second, the emulation software must generate virtual devices for each of the target application calls to the host operating system; and each of these virtual devices must provide calls to the actual host devices. Third, the emulator must treat all instructions as conservatively as it treats instructions which are directed to memory mapped I/O devices or risk generating exceptions from which it cannot recover. Finally, the emulator must maintain the correct target state at all times and store operations must always check ahead to determine whether a store is to the target code area. All of these requirements eliminate the ability of the emulator to practice significant optimization of the code run on the host processor and make this type of emulation much slower than running the target application on a target processor. Emulation rates less than one-quarter as fast as state of the art processors are considered very good. In general, this has relegated this type of emulation software to uses where the capability of running applications

designed for another processor is useful but not primary.

Regarding claims 6, 19 and 31

Cmelik et al teaches,

generating, from sequence of instructions, a sequence of generated instructions, a predetermined generated instruction being generated for each instruction in the sequence of instructions (column 4, lines 57-67," the host RISC processor and the hardware devices associated with it in a host RISC computer are usually quite different than are the devices associated with the processor for which the target application was designed; and the various instructions provided by the target application program are designed to cooperate with the device drivers of the target operating system in accessing the various portions of the target computer. Consequently, the emulation program, which changes the instructions of the target application program to primitive host instructions which the host operating system is capable of utilizing, must somehow link the operations designed to operate hardware devices in the target computer to operations which hardware devices of the host system are capable of implementing. Often this requires the emulator software to create virtual devices which respond to the instructions of the target application to carry out operations which the host system is incapable of carrying out because the target devices are not those of the host computer. Sometimes the emulator is required to create links from these virtual devices through the host operating system to host hardware devices which are present but are addressed in a different manner by the host operating system.

Regarding claims 7-9 and 32-34

Cmelik et al teaches,

partitioning sequence of instructions into a number of instruction groups, each instruction group including one or more instructions (column 22, lines 43-65, This may be understood by referring to FIG. 3 which illustrates in block diagram form the general functional elements of the microprocessor 10 of the invention. When the morph host 12 executes a target program, it actually runs the translator portion of the code morphing software 11 which includes the only original un-translated host instructions which effectively run on the morph host 12. To the right in the figure is illustrated memory divided into a host portion including essentially the translator and the translation buffer 14 and a target portion including the target instructions and data, including the target operating system. The morph host hardware begins executing the translator by fetching host instructions from memory and placing those instructions in an instruction cache. The translator instructions generate a fetch of the first target instructions stored in the target portion of memory. Carrying out a target fetch causes the integer unit to look to the official target instruction pointer register for a first address of a target instruction. The first address is then accessed in the translation look-aside buffer 31 of the memory management unit 33. The memory management unit 33 includes hardware for paging and provides memory mapping facilities for the TLB 31. Presuming that the TLB 31 is correctly mapped so that it holds lookup data for the correct page of target memory, the target instruction pointer value is translated to the physical address of the target instruction. At this point, the condition of the bit (T bit) indicating whether a translation has been accomplished for the target instruction is detected, but the access is a read operation and no T bit exception will occur.

- generating predetermined generated instruction for one instruction in each of instruction group (column (column 4, lines 57-67," the host RISC processor and the hardware devices associated with it in a host RISC computer are usually quite different than are the devices associated with the processor for which the target application was designed; and the various instructions provided by the target application program are designed to cooperate with the device drivers of the target operating system in accessing the various portions of the target computer. Consequently, the emulation program, which changes the instructions of the target application program to primitive host instructions which the host operating system is capable of utilizing, must somehow link the operations designed to operate hardware devices in the target computer to operations which hardware devices of the host system are capable of implementing. Often this requires the emulator software to create virtual devices which respond to the instructions of the target application to carry out operations which the host system is incapable of carrying out because the target devices are not those of the host computer. Sometimes the emulator is required to create links from these virtual devices through the host operating system to host hardware devices which are present but are addressed in a different manner by the host

Regarding claims 10, 11, 23, 39 and 40

Cmelik et al teaches,

incrementing a coverage counter when the condition code of the predetermined generated instruction is satisfied to provide an indication that corresponding instruction will be executed (column 17,lines 45-67, Memory stores positioned between the head of the queue and the gate are already committed to memory, while those positioned between the gate of the queue and the tail are not yet committed to memory. Memory stores generated during execution of host

translations are placed in the store buffer 50 by the integer unit in the order generated during the execution of the host instructions by the morph host but are not allowed to be written to memory until a commit operation is encountered in a host instruction. Thus, as translations execute, the store operations are placed in the queue. Assuming these are the first stores so that no other stores are in the gated store buffer 50, both the head and gate pointers will point to the same position. As each store is executed, it is placed in the next position in the queue and the tail point is incremented to the next position (upward in the figure). This continues until a commit command is executed. This will normally happen when the translation of a set of target instructions has been completed without generating an exception or a error exit condition. When a translation has been executed by the morph host 12 without error, then the memory stores in the store buffer 50 generated during execution are moved together past the gate of the store buffer 50 (committed) and subsequently written to memory. In the embodiment illustrated, this is accomplished by copying the value in the register holding the tail pointer to the register holding the gate pointer.

Regarding claims 12, 25 and 41

Cmelick et al teaches,

replacing a preceding instruction in sequence of generated instructions with predetermined generated instruction having a condition code corresponding to preceding instruction (column 20, lines 23-32, When this happens, the first translated host instruction branches back to the main loop where the next set of target instructions designated by the target instruction pointer is searched for in the translation buffer 14, the host translation is fetched from the

cache; or the search in the translation buffer fails, and the target instructions are fetched from memory and translated. When this translated host instruction is cached in the translation buffer 14, its address replaces the main loop address in the branch instruction which ended the loop).

Regarding claims 20, 21, 22, 24 and 25

Cmelik et al teaches,

instruction generation logic is operable to partition sequence of instructions into a number of instruction groups, each instruction group including one or more instructions, and to generate predetermined generated instruction for one instruction in each of instruction group (column 25, lines 45-65, column 26, lines 1-34).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Anil Khatri whose telephone number is 571-272-3725. The examiner can normally be reached on M-F 8:30-5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

***

ANIL KHATRI
PRIMARY EXAMINER